



UNIVERSIDADE  
LUSÓFONA

# Sistema de Rega

Grupo 1, Turno 5ªfeira 12h00 – 14h00

Tomás Nave, a22208623

André Jesus, a22207061

Arquitetura de Computadores | LEI | 23/03/2023

Índice

[www.ulusofona.pt](http://www.ulusofona.pt)

Introdução.....	3
Descrição do problema.....	4
Fluxograma .....	5
Tabela de Transições .....	7
Implementação .....	9
Conclusões .....	13

# Introdução

## **Parte 1 – Microprocessador rudimentar**

Com este trabalho pretendíamos desenvolver um automatismo de um sistema de rega tendo em atenção a fatores de avaria de sensores, avaria de mecanismos como a válvula do sistema e a outros fatores terceiros ao sistema.

Para realizarmos este projeto de automação tivemos que construir um fluxograma, que tivesse em conta todos os imprevistos, uma tabela de estados com base no fluxograma que tivesse todos os testes que foram realizados e por fim a programação das ROMS do circuito em logisim de acordo com os resultados obtidos.

## **Parte 2 – Microprocessador básico**

Na segunda parte deste trabalho melhoramos o nosso automatismo passando de um microprocessador rudimentar para um microprocessador básico o que nos permitiu que retirássemos as limitações humanas que tínhamos ao utilizar um microprocessador rudimentar.

## **Parte 3 – Microcontrolador Comercial**

Na terceira parte deste trabalho tivemos como objetivo codificar o nosso automatismo, ou seja codificar o nosso fluxograma que já tínhamos da parte um em PIC16F628A. Tivemos que aprender a trabalhar no MPLAB e a passar o nosso automatismo para assembly

## Parte 1 – Microprocessador rudimentar

### Descrição do problema

O problema que pretendemos resolver com este trabalho é um sistema de rega automático. Neste sistema temos um depósito de água que possui dois sensores S1 e S2 estando o sensor S1 um metro abaixo de S2. Este tanque possui uma válvula que é aberta quando o sinal está a 1, e essa válvula possui logo abaixo um sensor SH2 que tem como objetivo detetar se está a sair água ou não.

O início de toda esta automação é o sensor SH1 que tem como objetivo detetar se a horta está regada ou não, se o sensor estiver com o sinal a 1 quer dizer que a horta está regada, se estiver a 0 quer dizer que não está regada e que precisa de ser regada, ou seja vai dar início à automação.

Mas antes do início da automação existe um sensor SC que é o sensor que deteta se está a chover, ou seja se estiver a chover estando o SC a 1 o automatismo não liga pois se está a chover não é preciso gastar água do tanque para regar as plantas.

Agora se a horta precisa de ser regada ou seja SH1 a 0, e se não está a chover aí sim começa o automatismo.

Para a horta ser regada o depósito precisa estar cheio pelo menos até à marca do S2 ou seja a marca do S2 é a quantidade de água que é preciso para uma rega. O depósito é abastecido através de um furo que possui um motor para puxar a água,

Este tem também um sensor MSA que indica se o furo tem água ou não. Ou seja no automatismo primeiro é verificado se o tanque tem água para uma rega ou seja se está na marca de S2, se assim estiver o automatismo continua sem o motor funcionar pois não é preciso abastecer o tanque para fazer uma rega.

Se o depósito estiver abaixo da marca de S2 ou seja se S2 estiver a 0 primeiro vai ser verificado se o furo tem água ou não através do sensor MSA:

- se o furo não tiver água o sistema alerta que não existe água no furo ou seja não vai ser possível fazer a rega.

- se o furo tiver água o automatismo continua com o motor a trabalhar ou seja vai entrando água no depósito do furo e vai saindo ao mesmo tempo pela válvula pois vai estar a regar ao mesmo tempo.

Por fim estando o motor a trabalhar ou não é aberta a válvula essa válvula tem um sensor SH2 e um timer. Este timer tem programado já de origem o tempo que a água demora a chegar da válvula ao sensor pois esse tempo não é imediato.

Este timer T2 serve para distinguir se a válvula está estragada ou se é o sensor que está estragado, por outras palavras, se o SH2 estiver a 0 ou seja se o sensor da válvula não tiver detetado água e o SH1 estiver a zero significa que o terreno não está regado e que a válvula está avariada(ST2-1|ST1-0|ST0-1), mas se SH1 estiver a 1 significa que a horta está regada e que o SH2 está avariado(ST2-1|ST1-1|ST0-0).

(Usámos ST2-1|ST1-1|ST0-0 para indicar que o sensor de humidade, SH2, está avariado)

**OUTPUTS:**

ST – Start timer

NC - Não usado

T2 – Timer2

M – Motor de tirar a água para o depósito

V – Válvula

ST2 ST1 ST0 - Status

**INPUTS:**

S1, S2 – Sensores de água no depósito

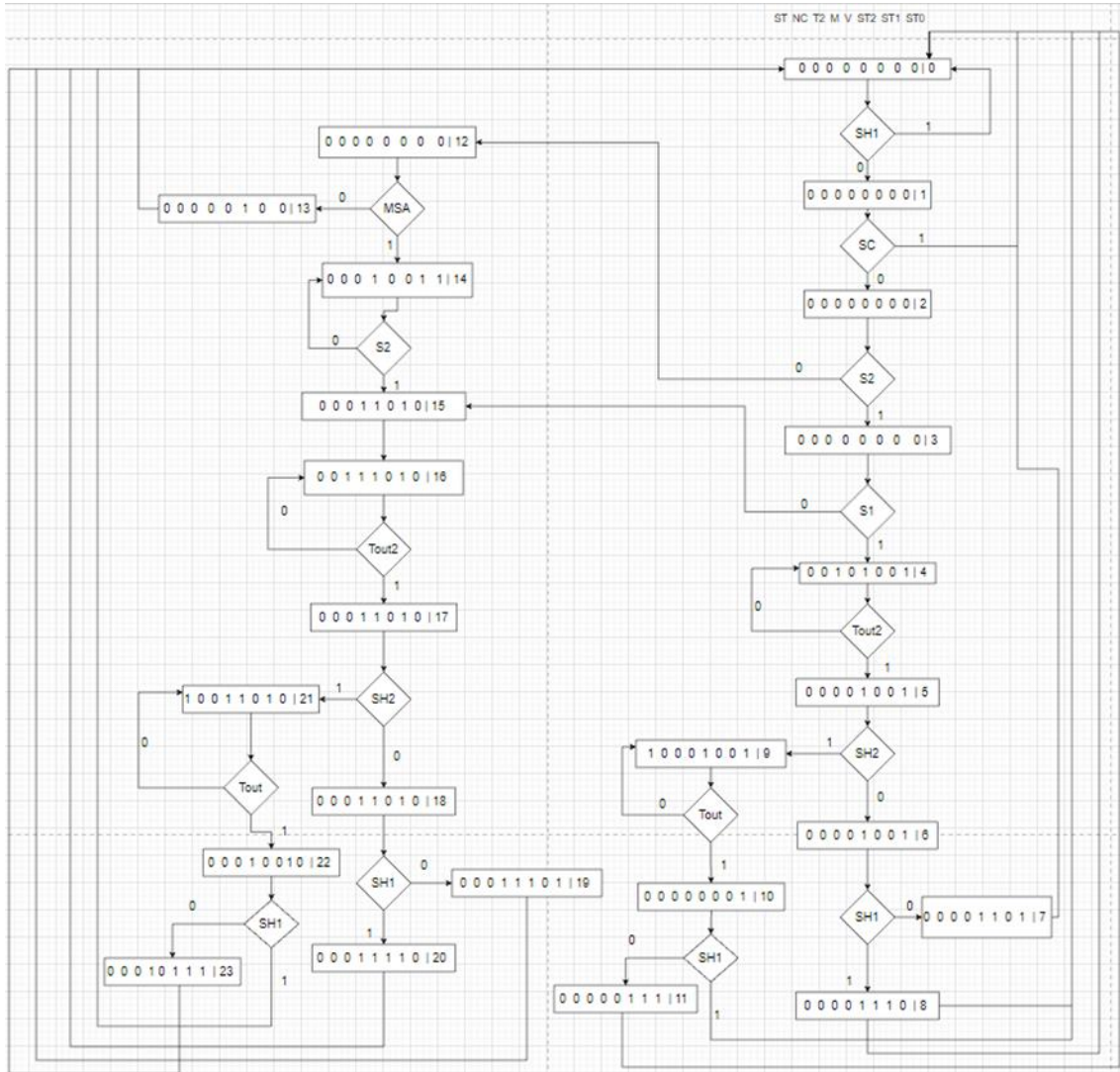
SH1, SH2 – Sensores de Humidade

MSA – Motor sem água no Furo

SC – Sensor de Chuva

TOUT, TOUT2 - Timerout

# Fluxograma



Começamos o fluxograma com a verificação se a horta está ou não regada, se estiver SH1(1) não é necessário regar a horta, se não estiver regada, SH1(0), temos de verificar se está ou não a chover, se estiver SC(1), não é necessário regar a horta, se não estiver a chover, SC(0), é preciso.

Se S2(1) então significa que temos água necessária para fazer uma rega.

Se S1(1) significa que o sistema de rega pode estar a trabalhar sem a ajuda do motor, pois não precisa de mais água no depósito, então ativamos um TOUT2 para que a água tenha tempo de chegar ao SH2, se SH2 se mantiver a 0, significa que algo correu mal, ou seja, ou a válvula está avariada e não deixa passar a água, ou o SH2 está avariado, para isso temos de verificar se o terreno foi ou não regado, se tiver sido regado então o SH2 está avariado, se o terreno estiver seco então a válvula está avariada.

Se SH2(1) então está a passar água por isso ativamos um TOUT para que dê tempo para a horta ficar regada, se ao fim do timer SH1(1) então a horta está regada, se SH(0) então houver uma falha grave e é necessário intervenção.

Voltando ao teste S2, se S2 estiver a 0 então temos que verificar se existe água no furo, através do sensor MSA, se não houver MSA(0) então não é possível regar a horta, se

MSA(1) então ligamos o motor e verificamos quando S2 fica ativo, quando estiver podemos começar a rega com o motor a trabalhar para que continue a puxar água para o depósito, de seguida efetuamos o mesmo processo já referido, ou seja ativamos um TOUT2 e quando este tiver terminado TOUT2(1) significa que a água já teve tempo suficiente para chegar ao SH2, se SH2 se mantiver a 0, significa que algo correu mal, ou seja, ou a válvula está avariada e não deixa passar a água, ou o SH2 está avariado, para isso temos de verificar se o terreno foi ou não regado, se tiver sido regado então o SH2 está avariado, se o terreno estiver seco então a válvula está avariada.

Se SH2(1) então está a passar água por isso ativamos um TOUT para que dê tempo para a horta ficar regada, se ao fim do timer SH1(1) então a horta está regada, se SH(0) então houver uma falha grave e é necessário intervenção.

## Tabela de Transições

Estados	Teste	ES0	ES1	OUTS
0	SH1	E1	E0	0 0 0 0 0 0 0 0
1	SC	E2	E0	0 0 0 0 0 0 0 0
2	S2	E12	E3	0 0 0 0 0 0 0 0
3	S1	E15	E4	0 0 0 0 0 0 0 0
4	TOUT2	E4	E5	0 0 1 0 1 0 0 1
5	SH2	E6	E9	0 0 0 0 1 0 0 1
6	SH1	E7	E8	0 0 0 0 1 0 0 1
7	XX	E0	E0	0 0 0 0 1 1 0 1
8	XX	E0	E0	0 0 0 0 1 1 1 0
9	MSA	E13	E14	0 0 0 0 0 0 0 0
10	XX	E0	E0	0 0 0 0 0 1 0 0
11	S2	E14	E15	0 0 0 1 0 0 1 1
12	XX	E16	E16	0 0 0 1 1 0 1 0
13	TOUT2	E16	E17	0 0 1 1 1 0 1 0
14	SH2	E18	E21	0 0 0 1 1 0 1 0
15	SH1	E19	E20	0 0 0 1 1 0 1 0
16	XX	E0	E0	0 0 0 1 1 1 0 1
17	XX	E0	E0	0 0 0 1 1 1 1 0
18	TOUT	E21	E22	1 0 0 1 1 0 1 0
19	SH1	E23	E0	0 0 0 1 0 0 1 0
20	XX	E0	E0	0 0 0 1 0 1 1 1
21	TOUT	E9	E10	1 0 0 0 1 0 0 1
22	SH1	E11	E0	0 0 0 0 0 0 0 1
23	XX	E0	E0	0 0 0 0 0 1 1 1

Por exemplo vamos ver o estado 5, realizando o teste SH2:

-se SH2 (estiver a 0) - Passa para o estado seguinte E6

-se SH2 (estiver a 1)- Passa para o estado seguinte E9

E o OUTS é o estado atual (0 0 0 0 1 0 0 1) , o 4 bite a 1 indicanos que a valvula está aberta e os ultimos 3 bites (0 0 1) indicam nos que está a regar com o motor desligado

-



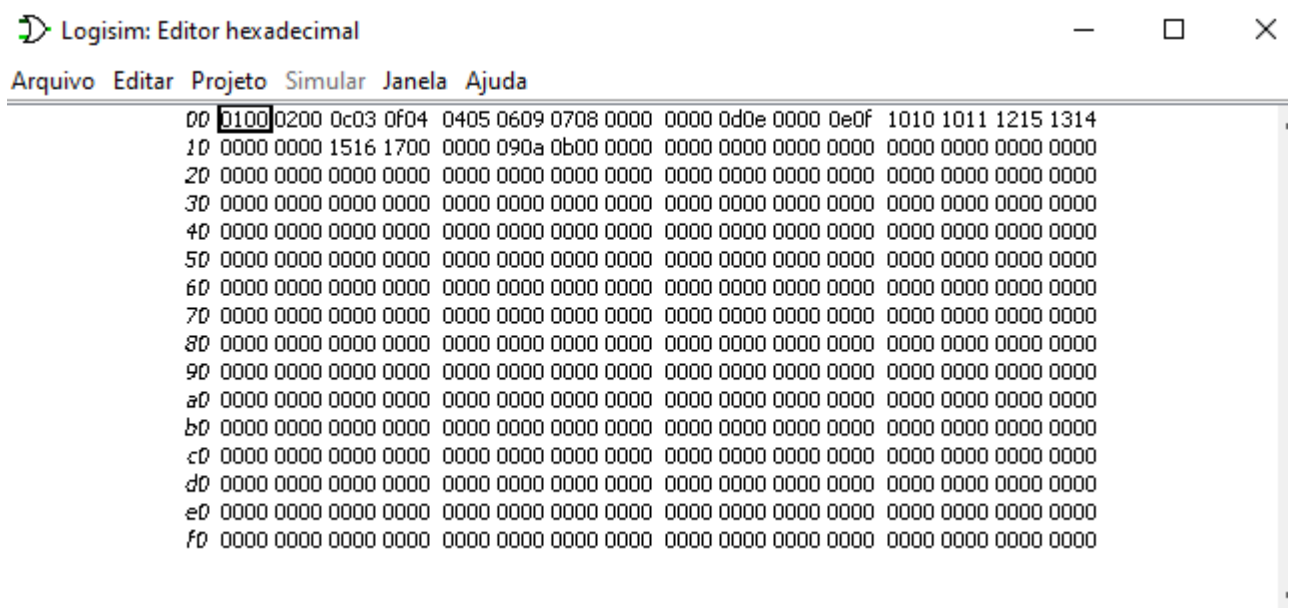
# Implementação

## Parte 1

O conteúdo que colocamos na ROM-ES foram todos os ES0 e ES1 por esta mesma ordem , na ROM-TST colocamos todos os testes efetuados, e por fim na ROM-OUT colocamos os respectivos OUTS.

Na nossa máquina implementamos dois timers,um relativo ao tempo a que a água demora a sair da Válvula até chegar ao Sensor de Humidade 2, e outro relativo ao tempo que demora o Sensor de Humidade 1 a detetar que a horta está regada.

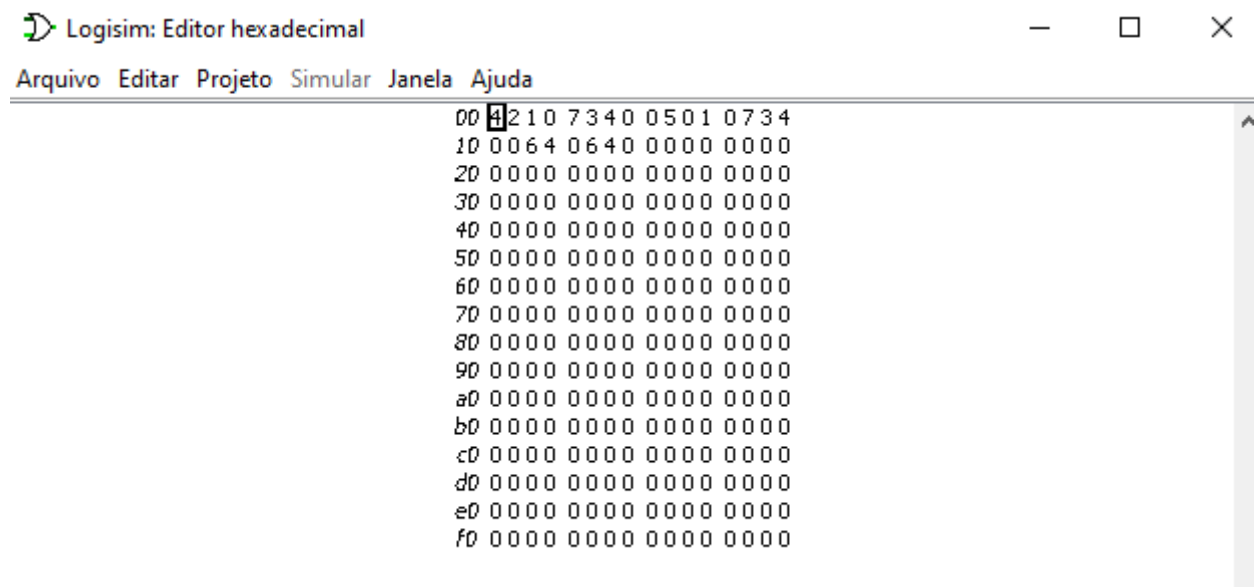
## ROM-ES:



The screenshot shows the Logisim hexadecimal editor window. The title bar reads "Logisim: Editor hexadecimal". The menu bar includes "Arquivo", "Editar", "Projeto", "Simular", "Janela", and "Ajuda". The main display area shows a memory dump for the ROM-ES. The first address, 00, is highlighted with a blue selection box and contains the value 0100. The subsequent addresses (10 to f0) all contain the value 0000. The memory dump is organized in columns of 16 hexadecimal digits per line, with addresses listed on the left.

```
00 0100 0200 0c03 0f04 0405 0609 0708 0000 0000 0d0e 0000 0e0f 1010 1011 1215 1314
10 0000 0000 1516 1700 0000 090a 0b00 0000 0000 0000 0000 0000 0000 0000 0000 0000
20 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
30 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
40 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
50 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
60 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
70 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
80 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
90 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
a0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
b0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
c0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
d0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
e0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
f0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

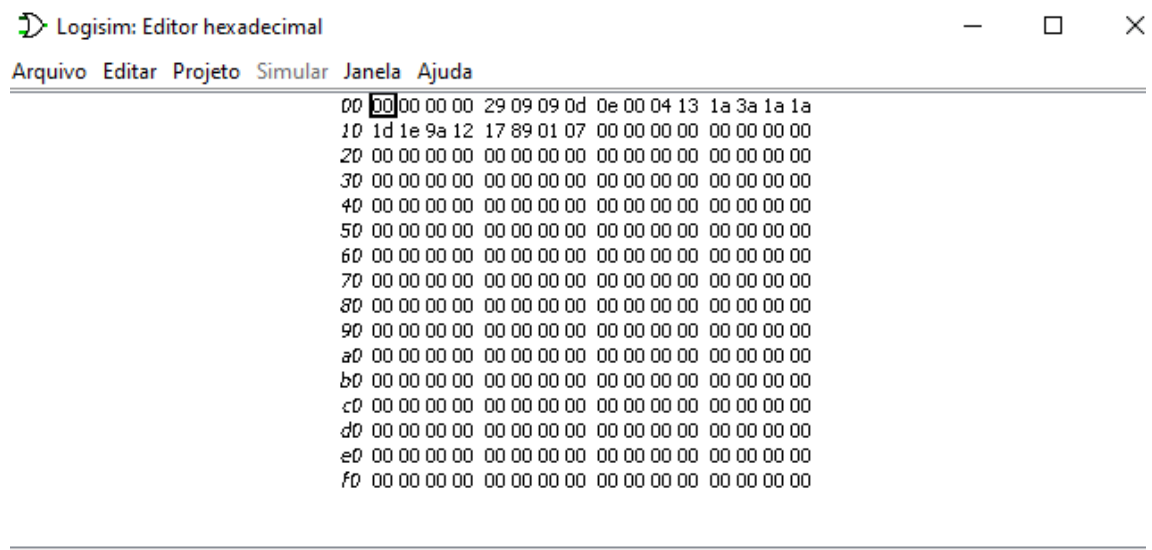
## ROM-TST:



The screenshot shows the 'Logisim: Editor hexadecimal' window. The title bar includes the application name and standard window controls (minimize, maximize, close). The menu bar contains 'Arquivo', 'Editar', 'Projeto', 'Simular', 'Janela', and 'Ajuda'. The main text area displays a list of hexadecimal addresses and their corresponding values. The address '00' is selected, and its value '210734005010734' is highlighted. The list continues with addresses from 10 to f0, all of which have the value '00000000'.

```
00 210734005010734
10 00000000
20 00000000
30 00000000
40 00000000
50 00000000
60 00000000
70 00000000
80 00000000
90 00000000
a0 00000000
b0 00000000
c0 00000000
d0 00000000
e0 00000000
f0 00000000
```

## ROM-OUT:



The screenshot shows the 'Logisim: Editor hexadecimal' window. The title bar includes the application name and standard window controls (minimize, maximize, close). The menu bar contains 'Arquivo', 'Editar', 'Projeto', 'Simular', 'Janela', and 'Ajuda'. The main text area displays a list of hexadecimal addresses and their corresponding values. The address '00' is selected, and its value '0000002909090d0e0004131a3a1a1a' is highlighted. The list continues with addresses from 10 to f0, all of which have the value '00000000'.

```
00 0000002909090d0e0004131a3a1a1a
10 0000001d1e9a12178901070000000000000000
20 00000000000000000000000000000000
30 00000000000000000000000000000000
40 00000000000000000000000000000000
50 00000000000000000000000000000000
60 00000000000000000000000000000000
70 00000000000000000000000000000000
80 00000000000000000000000000000000
90 00000000000000000000000000000000
a0 00000000000000000000000000000000
b0 00000000000000000000000000000000
c0 00000000000000000000000000000000
d0 00000000000000000000000000000000
e0 00000000000000000000000000000000
f0 00000000000000000000000000000000
```

## Parte 2 – Microprocessador básico

Utilizando o fluxograma do Trabalho 1 e a sua respetiva tabela de estados , fizemos o preenchimento da ROM da Memória do programa.

Como podemos ver nas tabelas abaixo a tabela da direita é a nossa tabela de estados do fluxograma.

A tabela da esquerda é a tabela que utiliza os valores da tabela do fluxograma e os coloca da estrutura necessaria para implementar na ROM do microprocessador básico.

Estrutura:

-Estado 0:

- Out - hexadecimal do out
- Test – hexadecimal do Test
- GOTOZ – Estado seguinte a 0 (Hex)
- GOTO – Estado seguinte a 1 (Hex)

A ROM memória vai ser preenchida de acordo com esta estrutura para cada estado, ou seja para cada estado vamos preencher 4 espaços da ROM.

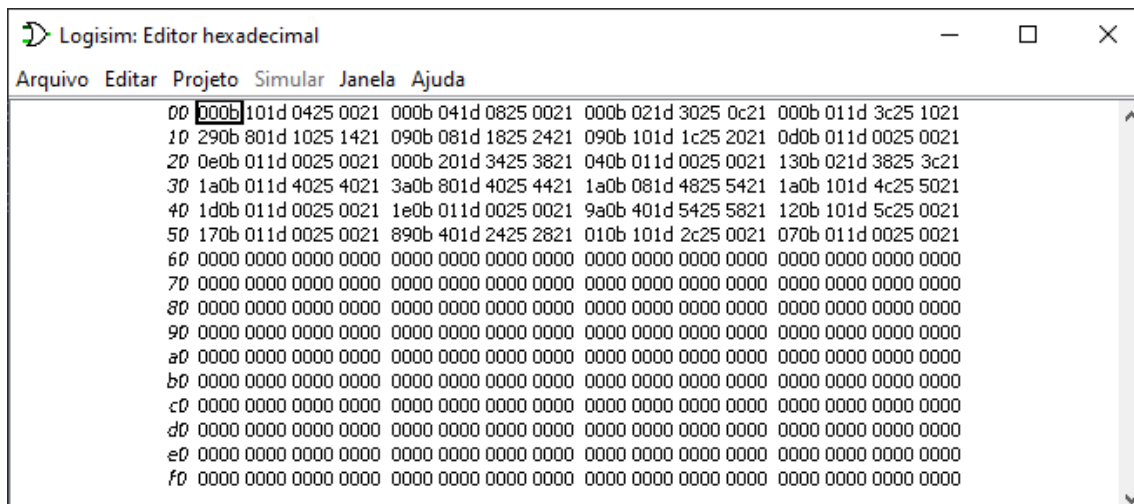
Estado (dec)	ES0 (dec)	ES1 (dec)	TESTE (bit)	Out (hex)
0	1	0	4	00
1	2	0	2	00
2	12	3	1	00
3	15	4	0	00
4	4	5	7	29
5	6	9	3	09
6	7	8	4	09
7	0	0	0	0d
8	0	0	0	0e
9	13	14	5	00
10	0	0	0	04
11	14	15	1	13
12	16	16	0	1a
13	16	17	7	3a
14	18	21	3	1a
15	19	20	4	1a
16	0	0	0	1d
17	0	0	0	1e
18	21	22	6	9a
19	23	0	4	12
20	0	0	0	17
21	9	10	6	89
22	11	0	4	01
23	0	0	0	07
24	0	0	0	00
25	0	0	0	00
26	0	0	0	00
27	0	0	0	00
28	0	0	0	00
29	0	0	0	00
30	0	0	0	00
31	0	0	0	00

Estado (dec)	ES0 (dec)	ES1 (dec)	TESTE (bit)	Out (hex)
0	00	00	000b	000b
1	01	10	101d	
2	02	04	0425	
3	03	00	0021	
4	04	00	000b	000b
5	05	04	041d	
6	06	08	0825	
7	07	00	0021	
8	08	00	000b	000b
9	09	02	021d	
10	0A	30	3025	
11	0B	0C	0C21	
12	0C	00	000b	000b
13	0D	01	011d	
14	0E	3C	3C25	
15	0F	10	1021	
16	10	29	290b	
17	11	80	801d	
18	12	10	1025	
19	13	14	1421	
20	14	09	090b	
21	15	08	081d	
22	16	18	1825	
23	17	24	2421	
24	18	09	090b	
25	19	10	101d	
26	1A	1C	1C25	
27	1B	20	2021	
28	1C	0d	0d0b	
29	1D	01	011d	
30	1E	00	0025	
31	1F	00	0021	
32	20	0e	0e0b	
33	21	01	011d	
34	22	00	0025	
35	23	00	0021	
36	24	00	000b	000b
37	25	20	201d	

- 1) Fazer copy da coluna da folha "HEX" para um ficheiro (\*.txt) em
- 2) Ler o Ficheiro no LogiSim para a ROM programa
- 3) Atencão ao formato das células exemplo

## MEM Programa:



## Parte 3 – Microprocessador Comercial em Assembly

Codificamos o nosso fluxograma da parte 1 do trabalho em PIC16F628A, onde tivemos que fazer a implementação do nosso fluxograma em Assembly.

Para fazer a implementação do fluxograma utilizamos a estrutura que está descrita na imagem apresentada à direita, de maneira a implementar-mos as variações de estados do nosso fluxograma.

Utilizamos:

PORTB: Para os INPUTS

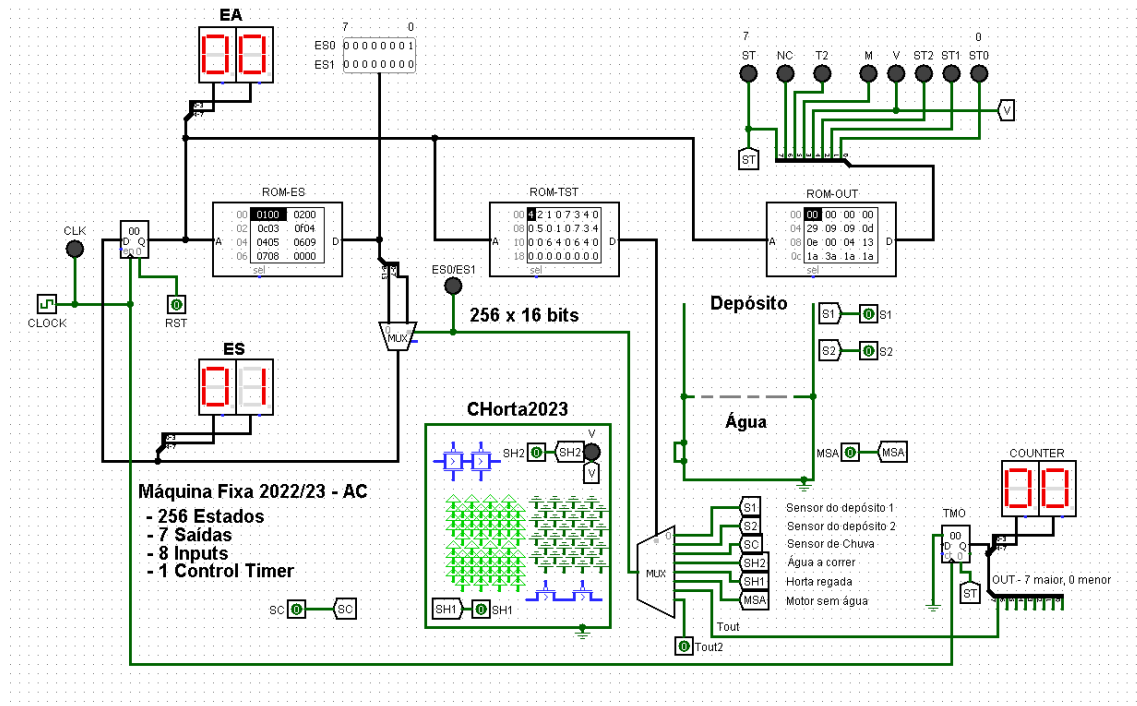
S1 (BIT 0)	S2 (BIT 1)
SC (BIT 2)	SH1 (BIT 3)
SH2 (BIT 4)	MSA (BIT 5)
TOUT2 (BIT 7)	TOUT (BIT 6)

PORTA: Para os OUTPUTS

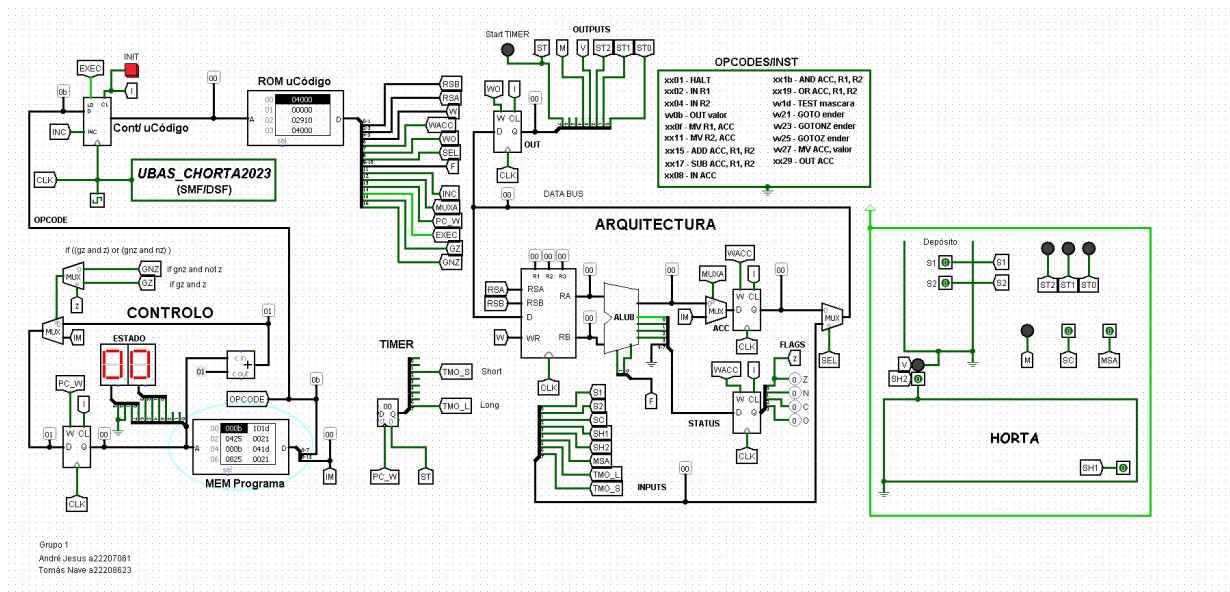
```

E0
    movlw 0x45
    movwf PORTB
    btfss PORTA, 1 ; test bit 1 (do INPUT= PORTA)
    goto E1      ; ES0
    goto E0      ; ES1
E1
    movlw 0x13
    movwf PORTB
    btfss PORTA, 0 ; test bit 0 (do INPUT= PORTA)
    goto E2      ; ES0
    goto E3      ; ES1
E2
    movlw 0x0a
    movwf PORTB
    btfss PORTA, 3 ; test bit 3 (do INPUT= PORTA)
    goto E1      ; ES0
    goto E0      ; ES1
E3
    movlw 0x00
    movwf PORTB ; no TESTE, go to next STATE
    goto E3     ; fica para sempre neste estado
    
```

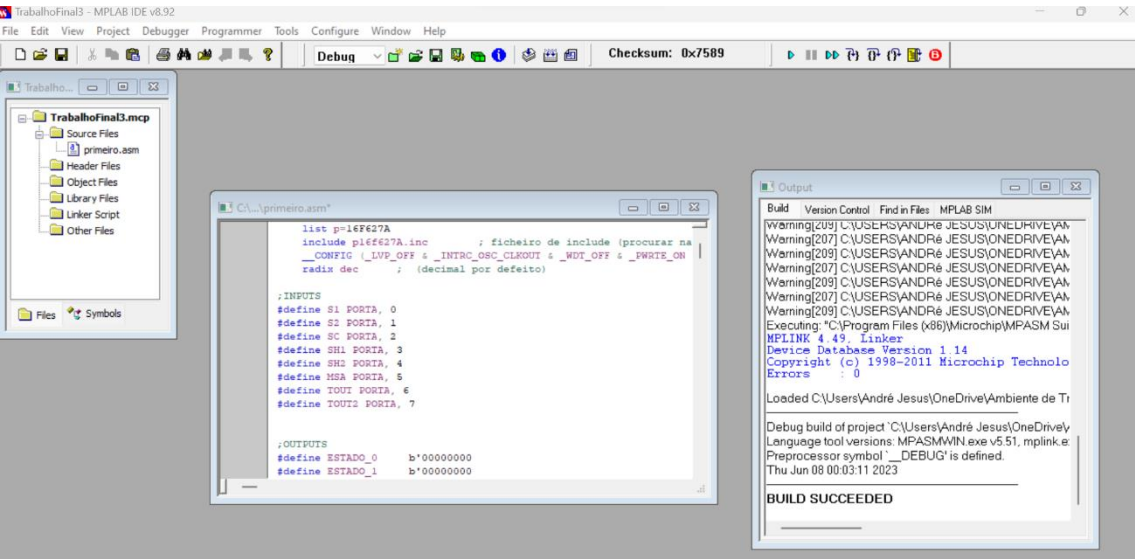
## Parte 1 – Microprocessador Rudimentar



## Parte 2 – Microprocessador Básico



Parte 3 – Microcontrolador Comercial



Link para descarregar a implementação no MPLAB:

[https://drive.google.com/file/d/11SmLcgDPbqMibJALQR74JNQmJM6S-c0p/view?usp=drive\\_link](https://drive.google.com/file/d/11SmLcgDPbqMibJALQR74JNQmJM6S-c0p/view?usp=drive_link)

Desvantagens e Vantagens dos difrentes metodos de codificacao

	Vantagens	Desvantagens
Máquina Fixa	Implementação mais simples	Tem muitas limitacoes pois precisa muito do ser humano para funcionar
	Mais facil de entender por ser menos complexo	
Microprocessador Rudimentar	A existencia de uma nova ROM para conseguirmos colocar os OPCODERS o que permitiu com que retirassemos as limitações humanas que tinhamos ao utilizar um microprocessador rudimentar	Não tem a possibilidade de fazer rotinas
		Se quisermos correr um segmento de código várias vezes em sítios diferentes no processador básico é necessário ter a sequência de instruções repetidas na memória de programa, ocupando mais memoria
Microcontrolador Comercial	A possibilidade da implementação de rotinas	Implementação mais dificil de entender devido á existencia de muitas operaçoes diferentes e instruções
	Tem mais memoria, pois possui um banco de registos que tem 4 RAMS dentro	Mais dificil de entender devido á linguagem Assembly

## **Conclusões**

A conclusão alcançada com este trabalho é que até um simples automatismo como um sistema de rega possui inúmeras condicionantes como cadeias enormes de funções que precisam de outras funções para só assim poderem ser realizadas, como enumeros fatores de avaria onde temos que prevelos e preparar o sistema para qualquer se seja o problema ou acontecimento, ou seja chegamos á conclusão que quando falamos de automações mesmo sendo para fazer uma função tao simples como regar um campo agricola a automação que está por trás é muito mais complexo do que aparenta.

### **Parte 1 – Parte 2**

O maior problema que sentimos durante a realização deste trabalho foi a implementação da automação no circuito no logisim devido a não estarmos totalmente confortáveis a trabalhar com ROMS.

### **Parte**

3

O maior problema que sentimos durante a realização deste trabalho foi a implementação do timer em assembly e a dificuldade em compreender a linguagem assembly visto que ainda não temos muita prática na mesma.